

UNITED STATES PATENT APPLICATION

FOR

A METHOD AND SYSTEM FOR PARALLEL HASH TRANSFORMATION  
FOR AN ADDRESS INPUT

Inventor:

Mark Gooch

A METHOD AND SYSTEM FOR PARALLEL HASH TRANSFORMATION  
FOR AN ADDRESS INPUT

5

TECHNICAL FIELD

10       The present invention relates generally to digital communication on networked digital computer systems and communication system networks. More specifically, the present invention pertains to address indexing and digital communications network protocols.

15       BACKGROUND ART

          The use of network based electronic communications and information processing systems for information control and information retrieval has rapidly proliferated in modern business environments. Within a typical enterprise, hundreds of client computer systems and server computer  
20       systems are constantly accessed by hundreds, or even thousands, of users for obtaining company information, news, competitive information, training materials, and the like, via one or more company wide LANs (local area networks) or WANs (wide area networks), or via the networked resources of the vast communications network known as the Internet.

25

          Generally, digital communications networks (e.g., LANs, WANs, the Internet, etc.) are packet switched digital communications networks. As used generally, the term network refers to a system that transmits any combination of voice, video and/or data between users. The network includes  
30       the underlying architecture of connected clients and servers and their associated software (e.g., network operating system in the client and server machines, the cables connecting them and the supporting hardware, such as hubs, switches, routers, etc.). Packet switching refers to subdividing data comprising a message into a number of smaller units of data, or packets, and

routing the packets individually through a number of nodes of the communications network.

5 The nodes of the digital communications network are generally made up of servers, clients, NOS (network operating system) services and supporting hardware. Servers are typically high-speed computer systems that hold programs and data or perform services that are shared by network users (e.g., the clients). The clients (e.g., desktop computer systems, workstations, and the like) are typically used to perform individualized, stand-  
10 alone processing and access the network servers as required. The actual communications path hardware is the cable (twisted pair, coax, optical fiber) that interconnects each network adapter. In wireless systems such as WLANs (wireless LANs) and the like, antennas, access point devices, and towers are also part of the network hardware.

15 Data communications within a network is generally managed by a one of a number of protocols such as, for example, TCP/IP, IPX, or the like. The physical transmission of data is typically performed by the access method (Ethernet, Token Ring, etc.) which is implemented in the network adapters  
20 that are plugged into the computer systems. The standardized communications protocols enable the widespread interoperability of communications networks and the widespread exchange of business related information.

25 In a large enterprise network or on the Internet, the Internet Protocol (IP) is used to route the packets among the various nodes or from network to network. Routers contain routing tables that move the datagrams (e.g., frames, packets, or the like) to the next "hop", which is either the destination network or another router. In this manner, packets can traverse several  
30 routers within an enterprise and a number of routers over the Internet.

Routers inspect the network portion (net ID) of the address and direct the incoming datagrams to the appropriate outgoing port for the next hop. Routers move packets from one hop to the next as they have routing  
35 information to indicate the most efficient path that a packet should take to

reach it's destination. Eventually, if the routing tables are correctly updated, the packets reach their destination. Routers use routing protocols to obtain current routing information about the networks and hosts that are directly connected to them.

5

In a manner similar to routers, many modern switches now include routing functionality. Such routing switches, as with routers, function by forwarding data packets from one local area network (LAN) or wide area network (WAN) to another. Based on routing tables and routing protocols, switches/routers read the network address in each transmitted frame and make a decision on how to send it based on the most expedient route (traffic load, line costs, speed, bad lines, etc.). These network addresses include both a MAC address (media access control address) and an IP address (Internet protocol address).

15

The routing tables are indexed with respect to the addresses of the various nodes of the communications network. These addresses are used to route the packets to the required destination. Since each component on the network has its address, the resulting address space can be extremely large and unwieldy. Large data spaces can be difficult to work with within high-speed router/switches. The problem is even more pronounced with the routers operating at the core of the extremely large networks many enterprises are building, and with routers and switches functioning near the core of the Internet. The resulting address space can span many hundreds of megabytes of memory. To manage the large address space, many prior art address space hashing schemes have been developed.

25

Address space hashing has become a widely used method to reduce the huge addressing space of a large network to a small, relatively inexpensive, memory table. Due to the fact that the majority of installed networks are based upon Ethernet protocols, many different types of Ethernet MAC address hashing-based address handling methods have been implemented. For example, when a packet arrives at a switch or router, it will need a destination address (DA) lookup to forward the packet, and possibly also a source address (SA) lookup to learn or authenticate the sending station. The

35

network addresses will be used to generate hashing pointers, which are normally around 10-20 bits depending on table size.

5 The hashing pointer is generated using a hash function, wherein a hash function H can be described as a transformation that takes a variable-size input m (e.g., 48-bit MAC SA/DA), and a variable-size key k, and returns a fixed-size hash value "h" (e.g., hashing pointer),  $h = H(m, k)$ . Each hashing pointer references a block of memory containing one or multiple MAC entries. Each entry stores the whole 48-bit MAC address and a switching tag related to this address. This entry contains information such as the next-hop forwarding data (the switch port(s) to forward the packet to, destination MAC address, destination VLAN, etc.), packet priority, etc.

15 When table referencing happens, the MAC address/addresses from the valid entry/entries under the hashing pointer will be compared against the original MAC address and a hit/miss or known/unknown decision will be made accordingly for the DA or SA lookup. Any further decisions based upon forwarding/learning etc., will be made based on the table search results and system setup. The goal of the system is to reduce the address size from a very large block (e.g., 48-bits or more) to a smaller more manageable block (e.g., 10-20 bits), while avoiding address aliasing, where two or more addresses generate a common hash pointer (e.g., a conflict or collision).

25 Hashing conflicts/collisions have a very adverse effect on the performance of the network router/switch. The hardware of the router/switch is optimized to perform the hashing address space translation very rapidly. In the event of a collision, either a new hash pointer is computed with a different key k (which consumes additional memory bandwidth) or a software based error handling routine is used to resolve the address aliasing. The software based routines execute much more slowly than the normal forwarding hardware. Thus, it becomes critical to network performance that the switch/router implement a fast and efficient address space hashing table.

35 One prior art solution to this problem involves use of an exceptionally large hashing pointer. For example, for a 48-bit input, a 24-bit hashing

pointer can be implemented as opposed to, for example, a smaller 10-bit hashing pointer. The 24-bit hashing pointer reduces the likelihood of collisions as addresses are transformed from 48 to 24-bits as opposed to 48 to 10-bits. Unfortunately, the 24-bit hashing pointer results in a larger routing table  
5 (e.g.,  $2^{24}$  number of entries) which requires more memory hence increases cost.

Another prior art solution is the use of a sophisticated hashing function for resolving the hash pointer. For example, a sophisticated hashing  
10 function can be designed to use each and every bit of a 48-bit input to generate a resulting 10-12-bit hashing pointer. The function can be configured to give a very high likelihood of different addresses transforming to different hashing pointers. Unfortunately, sophisticated and overly complicated hashing functions can be very difficult to implement in  
15 hardware. This can be even more problematic when the switch/router is designed to function at high-speed, wherein table lookups and routing decisions have to be made within a very small number of clock cycles.

Both of the above prior art solutions are increasingly outmoded, as the  
20 address spaces which are required to be efficiently indexed and tabled grow increasingly large. For example, newer versions of the Internet protocol (e.g., IPv6) will use 128-bit IP addresses. Thus, prior art type sophisticated hashing functions designed to use each and every bit of a 128-bit input to generate a hashing pointer become extremely difficult to implement using  
25 high-speed hardware. Similarly, prior art techniques using relatively large hashing pointers with respect to a 128-bit input require too much memory to implement cost-effectively.

Thus, the prior art is problematic in that conventional address space  
30 hashing schemes have difficulty scaling efficiently to large address spaces. Prior art address space hashing schemes have difficulty transforming input addresses into hashing pointers at high speed without increasing the number of conflicts/collisions which occur. Additionally, prior art address space hashing schemes that may have sufficient conflict/collision performance are

100202163-1

difficult to efficiently implement in high-speed hardware. The present invention provides a novel solution to these problems.

DISCLOSURE OF THE INVENTION

A method for performing a parallel hash transformation in a network device to generate a hash pointer for an address input is disclosed. In one embodiment, the method includes the step of receiving an address input. The address input is apportioned among a plurality of hashing units. The hashing units are configured to operate in parallel. A hash transformation is executed on the apportioned address inputs in parallel, resulting in a corresponding plurality of hashing unit outputs. The hashing unit outputs are combined to generate a hash result corresponding to the address input.

10



**BRIEF DESCRIPTION OF THE DRAWINGS**

The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention:

5

Figure 1 shows a diagram of a network system in accordance with one embodiment of the present invention.

Figure 2 shows a diagram of a 48-bit MAC destination address and a 48-bit MAC source address in accordance with one embodiment of the present invention.

10

15

Figure 3 a diagram of a 48-bit MAC destination address and a 48-bit MAC source address, in conjunction with a 32-bit source IP address and a 32-bit destination IP address in accordance with one embodiment of the present invention.

20

Figure 4 shows a diagram of a 128-bit source IP address and 128-bit destination IP address in accordance with one embodiment of the present invention.

25

Figure 5 a diagram of a parallel hashing system in accordance with one embodiment of the present invention.

Figure 6 shows a pseudo code representation of the logic function performed by a result combination unit in accordance with one embodiment of the present invention.

30

Figure 7 shows a routing table of 12-bit hash pointers generated from a 48-bit address input in accordance with one embodiment of the present invention.

35

Figure 8 shows a routing table of 12-bit hash pointers generated from a 32-bit IP address input in accordance with one embodiment of the present invention.

Figure 9 shows a routing table of 20-bit hash pointers generated from a 128-bit IP address input in accordance with one embodiment of the present invention.

5

Figure 10 shows a flowchart of the steps of a parallel hash generation process in accordance with one embodiment of the present invention.

BEST MODES FOR CARRYING OUT THE INVENTION

Embodiments of the present invention provide an address space hashing solution that can scale effectively to large address spaces. In addition, embodiments of the present invention implement an address space  
5 hashing method and system that transforms input addresses into hashing pointers while reducing the number of conflicts/collisions which occur. Furthermore, the address space hashing method and system can be efficiently implemented in high-speed hardware.

10 Referring now to Figure 1, a network system 100 in accordance with one embodiment the present invention is shown. As depicted in Figure 1, system 100 shows a router 120 and a switch 110 coupled to the Internet 130, in conjunction with router 140 and switch 150. These components  
15 comprise nodes which perform packet forwarding in accordance with embodiments of the present invention.

Referring still to Figure 1, a client 101 and a client 102 are coupled to switch 110 to receive and transmit information to the network of system 100. Packets from client 101-102 are transmitted and received through  
20 switch 110 and via router 120. Based upon the address of the packets (e.g., MAC addresses, IP addresses), the router 120 routes them to and from their required destination. For example, packets from client 101 can be routed to wireless access port 121 for communication with mobile users, or can be routed to switch 122 for communication with other clients connected to  
25 switch 122, etc. Similarly, packets from client 101 can be routed to other destinations across the Internet 130, such as, for example, a server 151 or client 152. In this example, the packets would be forwarded through router 140, switch 150, and into server 151.

30 Accordingly, router 120 and router 140 perform packet and/or frame router functions (e.g., forwarding data packets or frames from one local area network (LAN) or wide area network (WAN) to another). Router 120 and router 140 maintain internal routing tables, which, in conjunction with standardized routing protocols, allow for storing the network address in each

transmitted frame and make a decision on how to send it based on the most expedient route (traffic load, line costs, speed, bad lines, etc.).

In the present embodiment, routers 120 and 140, and switches 110, 122, and 150 are generally specialized hardware that is optimized for packet switched communications. However, this functionality can also be implemented in software on a general purpose computer having the necessary LAN and/or WAN interface(s).

Referring still to Figure 1, router 120 functions by examining the packets coming from client 101 to determine the routing port for transmitting packets to and from client 101. In determining the routing port, the router 120 will perform a destination address (DA) lookup to forward the packet, and may also perform a source address (SA) lookup to learn or authenticate the sending client, in this case client 101. In accordance with embodiments of the present invention, router 120 will use the destination IP address to generate a hashing pointer and use this hashing pointer to reference its internal hashing table. Each hashing pointer references a block of memory containing one or multiple IP entries (e.g., addresses). The entries are configured to map to the ports of the router 120 and are used by the router 120 to determine which port to forward the packet through.

In the present embodiment, to speed the routing process for the packets, router 120 implements a method for performing parallel hash transformations on the MAC SA/DA addresses and on the IP source and destination addresses received from the various connected clients. The hash transformations are used to generate a hash pointer for each address input.

Figure 2 shows a diagram of a 48-bit MAC destination address 201 and a 48-bit MAC source address 202 as operated on by embodiments of the present invention. As depicted in Figure 2, the 48-bit MAC SA/DA comprise the header of the data packet, referred to as a frame in an Ethernet based network. As known by those skilled in the art, the 48-bit MAC SA/DA comprise the hardware addresses of the various nodes connected to the network. For example, the network interface card of client 101 and client 102

(shown in Figure 1) will have their own respective MAC addresses. These addresses are used by switch 110 to determine where to forward the Ethernet frames.

5           Figure 3 shows a diagram of a 48-bit MAC destination address 301 and a 48-bit MAC source address 302, in conjunction with a 32-bit source IP address 310 and a 32-bit destination IP address 320, as operated on by embodiments of the present invention. Figure 3 shows a case where an Ethernet network (e.g., using layer 2 MAC SA/DA addresses) is used to support TCP/IP networking protocols, wherein sources and destinations are specified using IP addresses 310 and 320. As shown in Figure 3, the IP addresses 310 and 320 are included within the frame after the MAC SA/DA addresses 201 and 202.

15           Figure 4 shows a diagram of a 128-bit source IP address 410 and 128-bit destination IP address 420 as operated on by embodiments of the present invention. In this case, the frame is in accordance with version 6 of Internet protocols (e.g., IP version 6, or IPv6). In order to expand the number of Internet addresses available, as known by those skilled in the art, IPv6 addresses are 128-bits in length. In all other respects, frames are transmitted between network nodes in the same manner as for Ipv4 addresses.

25           Figure 5 shows a diagram of a parallel hashing system 500 in accordance with one embodiment of the present invention. As depicted in Figure 5, the hashing systems 500 includes four parallel hash units 520-523 configured to execute hash transformations on an address input in parallel. In this embodiment, system 500 operates on an input address of up to 128 bits (e.g., an IPv6 address input).

30           The hashing system 500 is implemented within network devices, for example, switch 110, router 120, router 140, and the like, in order to perform high-speed efficient forwarding in accordance with the present invention. The parallel hash units 520-523 are coupled to receive respective portions 510-513 of the 128-bit data input 501 as shown. The hash units 520-523 execute

their respective hash transformations in parallel to generate the resulting outputs 530-533. The outputs 530-533 are received by a result combination unit 540 which functions by recombining the outputs to obtain a 20-bit hash result 560 as shown.

5

In this manner, system 500 of the present embodiment provides an address space hashing solution that can scale effectively to large address spaces. By breaking down the hash generation into parallel execution units 520-523, the hash transformations can be performed on the large address input (e.g., 128-bit) much more quickly than attempting to perform a single monolithic hash transformation on the large address input using a single large hash unit. Parallel execution is much faster.

The parallel hash transformation of the present embodiment also provides an address space hashing solution that transforms input addresses into hashing pointers while reducing the number of conflicts/collisions which occur. Dividing the large address input into multiple hash unit inputs allows for more sophisticated hash algorithms to be implemented, whereby the hash result 560 is influenced by a greater proportion of the bits of the address input 501. Parallel execution allows more logical operations to be run on the bits comprising the address input 501.

Yet another advantage provided by system 500 of the present embodiment is the fact that parallel hash unit execution can be efficiently implemented in high-speed hardware. By dividing the large address input 501 amongst multiple hash execution units 520-523, the number of logic gates which a signal must cascade through remains limited in comparison with, for example, prior art monolithic, non-parallel schemes. Consequently, signal propagation through the parallel hash execution units of system 500 allow for very high-speed operation. For example, embodiments in accordance with system 500 can generate the hash result 560 from the address input 501 within a single clock cycle (e.g., 5 ns or less). Accordingly, system 500 can be integrated into a single ASIC.

It should be noted that system 500 of the present embodiment can be implemented, in whole or in part, in software executing on one or more digital processors. For example, the parallel execution units 520-523 can be implemented as parallel execution threads which can be distributed across parallel processors of a computer system. In such embodiment, it is desired that the hash execution is implemented in parallel, and as such, software for doing so can be distributed amongst computer system platforms or amongst processors of a single computer system.

Referring still to Figure 5, in the present embodiment, the result storage register 550 functions by allowing system 500 to accept successive 128-bit address inputs and generate a corresponding hash result 560. The result storage register 550 thus allows system 500 to operate on even wider address inputs (e.g., 256 bit). The result storage register 550 utilizes a recirculate result path 551, in conjunction with a control input 541, to recombine a previous result with a next result. This allows system 500 to perform hash transformations on wider inputs over multiple clock cycles. For example, system 500 can take a 128-bit address input and generate a resulting 20-bit hash result in one clock cycle, or alternatively, using the result storage register 550, system 500 can take 256 bits of address input data over two clock cycles and combined them into a single 20-bit result. The pseudocode for the result combination unit is shown in Figure 6.

Figure 6 shows a pseudo code representation of the result combination unit 540 of Figure 5. As shown in Figure 6, when control input 541 is assigned to zero, the recirculate result 551 is disabled, and the resulting output 560 is an XOR of the outputs of the hash units 520-523 (e.g., R3, R2, R1, and R0). When control input 541 is assigned to one, the recirculate result 551 is enabled, and the resulting output 560 is an XOR of the outputs of the hash units 520-523 and the recirculate result Rout 560 (e.g., R3, R2, R1, R0, and Rout).

It should be noted that embodiments of the present invention can use other types of logic to implement the hash functions of the hash units 520-523 besides XOR. The parallel hash transformation aspect of the present

invention provides enough performance margin to implement more complex hash functions.

Additionally, it should be noted that embodiments of the present invention can operate in conjunction with other types of address inputs besides 128-bit address inputs. For example, embodiments of the present invention can operate with address data that is less than 128 bits, with the unused bits being set to 0. Examples include 48-bit MAC addresses, 32-bit IPv4 addresses, and the like. Similarly, embodiments of the present invention can utilize greater or lesser degrees of parallel execution. For example, also system 500 utilizes four parallel hash units, other numbers of parallel hash units can be implemented (e.g., 2, 8, 10, etc.).

Figure 7, Figure 8, and Figure 9 show routing tables generated by systems in accordance with the present invention. Figure 7 shows a routing table 700 of 12-bit hash pointers generated from a 48-bit address input. As depicted in Figure 7, the routing table 700 has  $2^{12}$  entries for 48-bit MAC addresses as used in layer 2 switching.

Figure 8 shows a routing table 800 of 12-bit hash pointers generated from a 32-bit IP address input. As with routing table 700 of Figure 7, routing table 800 has  $2^{12}$  entries for 32-bit IP addresses used in layer 3 IP routing.

Figure 9 shows a routing table 900 of 20-bit hash pointers generated from a 128-bit IP address input. As depicted in Figure 9, the routing table 900 has  $2^{20}$  entries for 128 IP addresses used in the IPv6 routing protocols (layer 3).

Figure 10 shows a flowchart of the steps of a process 1000 in accordance with one embodiment of the present invention. As depicted in Figure 10, process 1000 shows the operating steps involved in a parallel hash transformation operation as performed when transforming packet address inputs into hash pointer outputs.



Process 1000 begins in step 1001 where an address input (e.g., 128-bit address input, etc.) is received at a hash transformation component of a network device. The hash transformation component functions by generating hash results from the address inputs. The hash results, in this case hash pointers to a forwarding table, are used by the network device (e.g., a switch or router) to forward packets or frames along the network. In step 1002, the address input is divided amongst a plurality of hashing units. For example, as described above, in a case where four parallel hash units are used to process a 128-bit address input, the 128-bit address input is divided into respective 32-bit inputs for the hash units. In step 1003, the hash units execute a hash transformation on the divided address inputs in parallel. Subsequently, in step 1004, the hashing unit outputs are combined to generate a hash result corresponding to the address input.

Thus, a method and system for parallel hash transformation for an address input has been disclosed. The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order best to explain the principles of the invention and its practical application, thereby to enable others skilled in the art best to use the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.